2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Dynamic Boosting in Deep Learning using Reconstruction Error

Wenhao Huang, Ni Zhang, Weisong Hu, Haikun Hong, Guojie Song, Kunqing Xie

Abstract— **Deep learning has attracted a lot of attention in research and industry in recent years. Behind the success of deep learning, there is much space for improvement. It is difficult to identify if a testing sample can be represented by the deep network effectively before we examining the final result. In this paper, we proposed a dynamic boosting strategy according to reconstruction error in deep networks. We use reconstruction error to determine whether the result is reliable or not. From the perspective of prediction interval, we demonstrated that with the increase of reconstruction error, the prediction interval would become bigger. Therefore, the classification result is not reliable when the reconstruction error exceeds the pre-determined threshold. Since we can record the reconstruction error as well as the classification error for all training samples in training set. We can learn an extra boosting model besides the deep network in training set to improve the performance of the model. An important factor in learning the boosting model is to determine an appropriate threshold for selecting training samples. In testing, we first examine whether the reconstruction error of a testing sample exceeds the threshold to determine if we should use the boosting model. If the boosting model is used, the final result is the average of the output of the deep network and the boosting model. We conducted experiments on two widely used classification datasets and an air quality dataset. From the experiments, we see that our boosting strategy is effective in improving the performance of classification. We tested several boosting models in this paper. They can all reduce the test error to some extent under appropriate parameter settings.**

I. Introduction

Deep learning has attracted a lot of attention in research and industry in recent years [10][1][13]. Due to its strong ability of unsupervised feature learning, deep learning has achieved several state-of-the-art results in image processing, speech recognition and natural language processing. The main claim behind the deep network can be summarized as: several hidden layers of feature representation, stacked on the top of each other, are effective in learning better feature representation than single hidden layer [10]. Though the compactness and expressiveness of deep architectures had been known for decades, the bottleneck of training deep architectures has been overcame in recent years through layer-wise training in [9][2]. The deep learning approach aims at greedy optimization of reconstruction error in pre-training of each hidden layer to learn the most representative features. It is claimed that the best feature representation could reconstruct the original data with the least error.

Wenhao Huang, Haikun Hong, Guojie Song, Kunqing Xie are with the Department of Electronic Engineering and Computer Science, Peking University, China.

Ni Zhang and Weisong Hu are with the NEC Laboratories, China.

Guojie Song is the corresponding author of this paper (corresponding email:gjsong@pku.edu.cn).

Behind the success of deep learning, there is much space for improvement [6]. Like other machine learning approaches, the feature representation obtained from deep learning could only represent the majority of samples in training data. Therefore, it is difficult to identify if a testing sample can be represented by the model effectively before we examining the final result. A common way is investigating the data distribution to see if a testing sample is suitable for the model. Reconstruction error in deep learning provided us a promising way of determining whether the model is suitable for representing a testing sample. The final result is reliable if the reconstruction error is small since the model could learn good feature representation for the testing sample while the result is unreliable if the model could not represent the data very well.

Another advantage of using reconstruction error is that it is very convenient to obtain the reconstruction error and the output error of the training data [16]. It is possible to analysis the relationship between the reconstruction error and the output error of training data to guide the supervised task for a testing sample. With proper strategies, we could build a boosting model according to the reconstruction error of the training data.

In this paper, we first investigate the relationship between the reconstruction error and the error of supervised learning. From the perspective of prediction interval, we demonstrate that the prediction interval would be bigger with the increase of the reconstruction error. In other words, the result produced by the model becomes less reliable when the reconstruction error increases. Then we propose a dynamic boosting strategy to improve the performance of the model based on reconstruction error. For a classification task, an extra boosting model is used when the reconstruction error exceeds the threshold. We first compute reconstruction error and classification error for all training samples by the deep network. Classification error is the differences of probability of each class obtained by first deep network and real class label. Then we select part of the training samples by the threshold of overall reconstruction error as training set for the boosting model. A number of existing classification models are implemented as the boosting model. We also built a dual deep network model for boosting. One deep network uses the data as input and class label as output. The other deep network uses the reconstruction error as input. Finally, we conducted abundant experiments to validate the effectiveness of the boosting strategy. Experimental results demonstrate that using reconstruction error is effective in improving performance of deep networks.

The rest of this paper is organized as follows. Section II introduces backgrounds of deep learning and reconstruction error. Section III investigates the relationship of reconstruction error and classification error. The dynamic boosting strategy is introduced in Section IV. Experiments are reported in section V. Conclusion and future prospects are given in Section VI.

II. Backgrounds

Before introducing our approach, we first review backgrounds of deep learning and reconstruction error in this section.

A. Deep Learning

Recent works on deep learning have demonstrated that deep sigmoidal networks could be trained layer-wise to produce good results for many tasks such as image and audio classification [13][9][15].

Deep Belief Network is the most effective approach among all deep learning models. It is a stack of Restricted Boltzmann Machines each having only one hidden layer. The learned units activations of one RBM are used as the "data" for the next RBM in the stack. Hinton et al. proposed a way to perform fast greedy learning of the deep network [9].

An RBM is a particular type of Markov Random Fields (MRF). It is an undirected graphical model in which visible variables ($\boldsymbol{v}$) are connected to stochastic hidden units ($\boldsymbol{h}$) using undirected weighted connections [21]. They are restricted that there are no connections within hidden variables or visible variables. The model defines a probability distribution over $\boldsymbol{v}, \boldsymbol{h}$ via an energy function. Suppose it is a binary RBM, it could be written as:

$$\begin{aligned} & -\log P(\boldsymbol{v}, \boldsymbol{h}) \propto E(\boldsymbol{v}, \boldsymbol{h}; \theta) \\ & = -\sum_{i=1}^{|V|} \sum_{j=1}^{|H|} w_{ij} v_i h_j - \sum_{i=1}^{|V|} b_i v_i - \sum_{j=1}^{|H|} a_j h_j \end{aligned} \tag{1}$$

where $\theta = (\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{a})$ are parameters, w_{ij} is the symmetric weight between visible unit i and hidden unit j while b_i and a_j are their bias. Number of visible and hidden units is represented as $|V|$ and $|H|$. This configuration makes it easy to compute the conditional probability distributions, when $\boldsymbol{v}$ or $\boldsymbol{h}$ is fixed.

$$\begin{aligned} p(h_j|\boldsymbol{v}; \theta) &= sigm(\sum_{i=1}^{|V|} w_{ij} v_i + a_j) \\ p(v_i|\boldsymbol{h}; \theta) &= sigm(\sum_{j=1}^{|H|} w_{ij} h_j + b_i) \end{aligned} \tag{2}$$

where $sigm(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function. The parameters of the model $\theta = (\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{a})$ could be learned using contrastive divergence [8] effectively.

Then we could stack several RBMs together into a DBN. The key idea behind training a DBN by training a series of RBMs is that parameters θ learned by an RBM define both $p(\boldsymbol{v}|\boldsymbol{h}, \theta)$ and prior distribution $p(\boldsymbol{h}|\theta)$ [17]. Therefore, probability of generating visible variables could be written as:

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\theta) p(v|h, \theta) \tag{3}$$

After θ is learned from an RBM, $p(\boldsymbol{v}|\boldsymbol{h}, \theta)$ is kept. In addition, $p(\boldsymbol{h}|\theta)$ could be replaced by consecutive RBM which treats hidden layer of previous RBM as visible data. By this way, it could improve a variational lower bound on the probability of the training data as introduced in [9].

Another frequently used deep neural network is deep auto-encoder [3][22][14]. Stacked auto-encoders use an auto-encoder instead of the RBM as a layer building block. An auto-encoder is composed of two parts: *encoder* and *decoder*.

Encoder: The encoder transforms an input vector $\boldsymbol{x}$ into hidden representation $\boldsymbol{x}$.

$$\boldsymbol{h} = f_\theta(\boldsymbol{x}) = sigm(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{4}$$

where $\theta = \boldsymbol{W}, \boldsymbol{b}$ is the parameter set of the encoder.

Decoder: The decoder maps the hidden representation $\boldsymbol{h}$ back to a reconstructed input vector $\boldsymbol{x'}$.

$$\boldsymbol{x'} = g_{\theta'}(\boldsymbol{h}) = sigm(\boldsymbol{W'}\boldsymbol{h} + \boldsymbol{b'}) \tag{5}$$

where $W' = W^T$ and b' is the bias for the decoder. The objective function of auto-encoder could be formulated as loss function between input vector $\boldsymbol{x}$ and reconstruction vector $\boldsymbol{x'}$, i.e. $L(\boldsymbol{x}, \boldsymbol{x'}) = |\boldsymbol{x} - \boldsymbol{x'}|^2$. For greedy layer-wise training, hidden representation $\boldsymbol{h}$ could be computed by optimizing the objective function of one layer. Then $\boldsymbol{h}$ is used as input vector for the next layer of the stacked deep auto-encoders.

B. Reconstruction Error

Here, we give an illustration of reconstruction error in deep networks in Figure 1. From a raw input data, the *encoder* from bottom to up is used to learn a good feature representation. The top feature representation (h_3 in the figure) is used for detail classification or regression task (Y). Meanwhile, the *decoder* from top to bottom can reconstruct the raw input data. The difference between the raw input data and the reconstruction data is referred to as **reconstruction error**. The objective function of auto-encoder is actually to minimize the reconstruction error to learn the most representative features in the top layer. As introduced above, we can learn an auto-encoder between two layers and then stack them together to a deep network.

It is worth to notice that the reconstruction error is at the same dimension as the input data. For each input data, we can obtain the reconstruction of the data from the deep network before we use the features learned from the model for classification task. Therefore, we can record reconstruction error of each sample in training data. Since we know the real class label (Y) for each sample in training data, we can obtain classification error as well. In deep network for classification, class label is represented as a vector $Y = \{y_1, y_2, \ldots, y_m\}$ where m is the number of classes. In the real class vector, $y_i = 1$ if the label is class i and the others $y_j = 0, j \neq i$. The

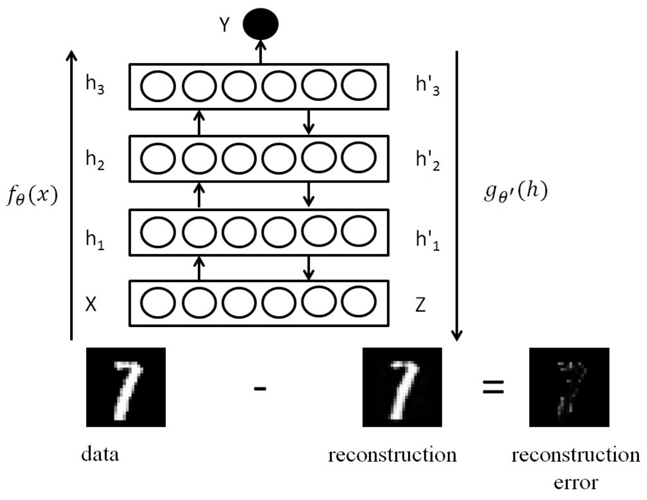


Fig. 1: Illustration of reconstruction error.

result from the model is also a vector $Y' = \{y'_1, y'_2, \ldots, y'_m\}$. The classification error is represented as $Y - Y'$.

III. Prediction Interval of Deep Network

In this section, we first give definition and computing method of prediction interval of neural networks. Since deep network is a special kind of neural network, the definition is almost the same. Then we analyze the relationship between the prediction interval of a deep network and the reconstruction error.

A. Prediction Interval of Neural Network

Concept of prediction interval used in this study is the same as the concept used in Shrestha et.al [20]. Figure 2 schematically demonstrated the definition of prediction interval [20]. For most of the models, it could give the prediction interval besides the model output from distribution of training data. From the prediction interval, we could estimate the reliability of the model output. If the prediction interval is big, the probability of the model output being unreliable is big as well. In other words, it is risky to use the model output as the result, though it is still possible that the error of model output is small while the prediction interval is big.

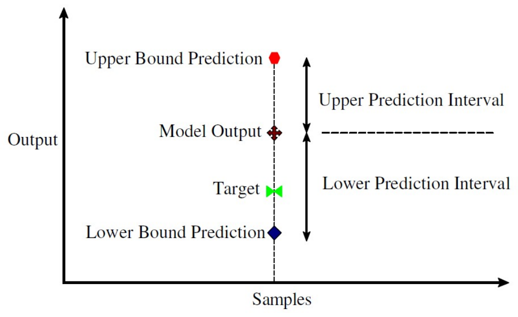


Fig. 2: Concept of prediction interval.

There are a lot of works focusing on prediction interval of neural networks [12][11]. Existing approaches can be divided into four categories: delta method [11][5], Bayesian method [4], mean-variance estimation (MVE) method [18], and bootstrap method [7]. Actually, the results obtained from these methods are similar. In this study, we use delta method to compute prediction interval in deep networks.

Here, we give a brief introduction of delta method. It is often assumed that targets can be modeled by:

$$t_i = y_i + \epsilon_i \tag{6}$$

where t_i is the i^{th} measured target, ϵ_i is the noise with a zero expectation, and y_i is the true regression value. Suppose the model output is $\hat{y}_i$, we have:

$$t_i - \hat{y}_i = (y_i - \hat{y}_i) + \epsilon_i \tag{7}$$

Consider that w^* is the set of the optimal parameters for the neural network, i.e., $y_i = f(x_i, w^*)$. In a small neighborhood of the optimal parameter set, the neural network model can be linearized as:

$$\hat{y_0} = f(x_0, w^*) + g_0^T(\hat{w} - w^*) \tag{8}$$

where x_0 is the input of a sample, $\hat{y_0}$ is the model output, g_0^T is the output gradient against the network parameters w^*, and $\hat{w}$ is the parameter set of the model. g_0^T can be formulated as:

$$g_0^T = [\frac{\partial f(x_0, w^*)}{\partial w_1^*}, \frac{\partial f(x_0, w^*)}{\partial w_2^*}, \ldots, \frac{\partial f(x_0, w^*)}{\partial w_p^*}] \tag{9}$$

where p is the number of parameters. According to this, we have:

$$\begin{aligned} t_0 - \hat{y_0} &\simeq (y_0 + \epsilon_0) - (f(x_0, w^*) + g_0^T(\hat{w} - w^*)) \\ &= \epsilon_0 + g_0^T(\hat{w} - w^*) \end{aligned} \tag{10}$$

and

$$var(t_0 - \hat{y_0}) = var(\epsilon_0) + var(g_0^T(\hat{w} - w^*)) \tag{11}$$

Assuming that the error terms are normally distributed, i.e., $\epsilon \approx N(0, \sigma_\epsilon^2)$. We can have:

$$var(g_0^T(\hat{w} - w^*)) = \sigma_\epsilon^2 g_0^T(F^T F)^{-1} g_0 \tag{12}$$

where σ indicates the variance, and F is the Jacobian matrix of the neural network model. It can be computed from training samples as follows:

$$F = \begin{bmatrix} \frac{\partial f(x_1,\hat{w})}{\partial(\hat{w}_1)} & \frac{\partial f(x_1,\hat{w})}{\partial(\hat{w}_2)} & \cdots & \frac{\partial f(x_1,\hat{w})}{\partial(\hat{w}_p)} \\ \frac{\partial f(x_2,\hat{w})}{\partial(\hat{w}_1)} & \frac{\partial f(x_2,\hat{w})}{\partial(\hat{w}_2)} & \cdots & \frac{\partial f(x_2,\hat{w})}{\partial(\hat{w}_p)} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f(x_n,\hat{w})}{\partial(\hat{w}_1)} & \frac{\partial f(x_n,\hat{w})}{\partial(\hat{w}_2)} & \cdots & \frac{\partial f(x_n,\hat{w})}{\partial(\hat{w}_p)} \end{bmatrix} \tag{13}$$

The total variance can be formulated as:

$$\sigma_0^2 = \sigma_\epsilon^2(1 + g_0^T(F^T F)^{-1} g_0) \tag{14}$$

An unbiased estimate of σ_ϵ^2 can be obtained by:

$$s_\epsilon^2 = \frac{1}{n-1} \sum_{i=1} n(t_i - \hat{y}_i)^2 \tag{15}$$

Finally, the prediction interval of $(1-\alpha)\%$ for $\hat{y_0}$ can be computed as:

$$\hat{y_0} \pm t_{n-p}^{1-\frac{\alpha}{2}} s_\epsilon \sqrt{1 + g_0^T (F^T F)^{-1} g_0} \tag{16}$$

where $t_{n-p}^{1-\frac{\alpha}{2}}$ is the $(\alpha/2)$ quantile of a cumulative t-distribution function with $n-p$ degrees of freedom.

B. Prediction Interval of Deep Network

As a special kind of neural network, we could formulate the prediction interval of deep network using the general form. The Jacobian matrix F and the unbiased estimate of σ_ϵ^2 (s_ϵ^2 in (15)) are all computed from training data. Therefore, for a sample in testing set $\{x_0, y_0\}$, the only relative term in the prediction interval is g_0.

From previous works of Hinton and Bengio, we can get that minimizing the reconstruction error in training a deep network would tight the variational lower bound on the log-probability of the testing data to ensure better performance in classification task. With that claim, we can get the following:

$$\begin{aligned} g_0 &\propto \frac{\partial}{\partial\theta}(log \sum_h \exp[-E(x_0, h|\theta)] - log \sum_x \sum_h \exp[-E(x, h|\theta)]) \\ &= (< \frac{\partial(-E(x_0, h|\theta))}{\partial\theta} >_{P(h|x_0)} - < \frac{\partial(-E(x, h|\theta))}{\partial\theta} >_{P(x,h|\theta)}) \\ &=< x_0 >_{data} - < x_0 >_{reconstruction} \\ &\approx x_0 - x_0'(\text{reconstruction error}) \end{aligned} \tag{17}$$

where θ is the parameters in the deep network and $< \cdot >_P$ is the expectation of the distribution P. The final step in the equation is approximated by contrastive divergence. By now, we can get that the prediction interval (PI_{x_0}) of a testing sample x_0 is proportional to the reconstruction error of the sample $(x_0 - x_0')$.

$$PI_{x_0} \propto (x_0 - x_0') \tag{18}$$

From the analysis above, we can see that reconstruction error is related with prediction interval. Small reconstruction error means a very tight prediction interval. It can ensure accurate classification result. The prediction interval is wide when the reconstruction error is big. Under this situation, it is possible that the final classification error is small. However, the probability of the classification error being big also increases. In other words, the result is not reliable when the reconstruction error is big. Therefore, we can tell whether the final result is reliable enough from the reconstruction error. It is also worth to notice that we can obtain the reconstruction error at the same time when we obtain the model output. So that we can use boosting strategies on model output to make it more reliable. Meanwhile, we can also obtain reconstruction error and classification error of all training samples as an extra dataset to train and learn an effective boosting strategy.

IV. Dynamic Boosting Strategy

In this section, we would introduce our dynamic boosting strategy. A boosting model is proposed for those samples whose reconstruction error exceeds the threshold. The boosting model is trained using part of the training samples selecting by the threshold.

A. Overview

We give an overview of the dynamic boosting strategy in Figure 3. The deep network can be a stacked Restricted Boltzmann Machine or a stacked auto-encoders. It can be trained using the traditional deep learning approach as introduced in [9][15]. We first train a deep network and then obtain reconstruction error of all the samples in the training set. The reconstruction error set is used to train the boosting model to produce another classification result.

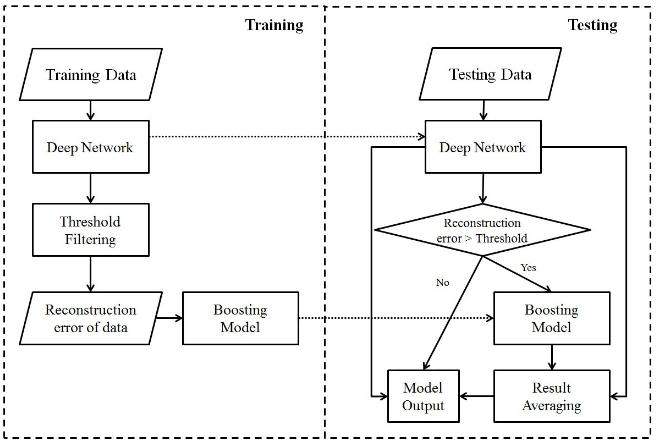


Fig. 3: Overview of the dynamic boosting strategy.

B. Deep Network

We first briefly introduce the deep network for classification. The architecture of the deep network is illustrated in Figure 4. The raw data is used as the input data X here. Then a deep network in the bottom is used to learn a feature representation in an unsupervised way. Using the deep network in the bottom, we can also obtain the reconstruction error $re = X - X'$. The top layer is for supervised classification. The result is represented as a probability vector of each class, i.e., $Y' = \{y_1', y_2', \ldots, y_m'\}$ where m is the number of classes. For supervised training, we are able to obtain the supervised classification error $se = Y - Y'$.

C. Boosting Model

An important factor to the boosting model is the filter criterion. As demonstrated in the section of prediction interval, we can see that the result is reliable when the reconstruction error is small. Therefore, the dynamic boosting strategy only pay attention to those samples whose reconstruction error is bigger than others. Since we can obtain reconstruction error of all training samples, we use a threshold δ to determine which part of the samples need to be retrained by the

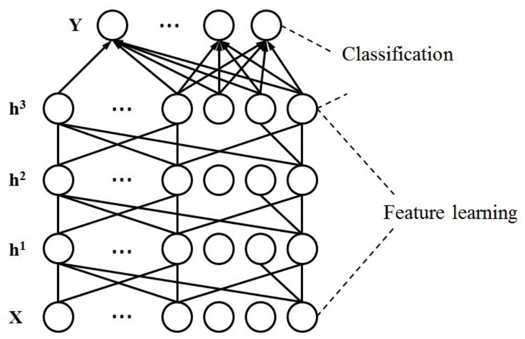


Fig. 4: Architecture of the deep network.

boosting model. There are some ways to define the value of overall reconstruction error. The first one ζ_1 is the sum of reconstruction error, denoted as:

$$\zeta_1 = \sum_{i=1}^{n} |x_i - x'_i| \tag{19}$$

This definition uses the sum of the reconstruction error to measure the reliability of the result.

The second one ζ_2 only takes the features with big reconstruction error into account.

$$\zeta_2 = \sum_{i \in top(|x_i - x'_i|)} |x_i - x'_i| \tag{20}$$

where $top(|x_i - x'_i|)$ selects the features with the top-k biggest reconstruction error. We may need another parameter k to measure the number of features. Actually, ζ_1 is a special case of ζ_2 where $k = n$.

The third one ζ_3 uses another threshold δ' to measure the reconstruction error of each feature x_i. It can be formulated as:

$$\zeta_3 = \sum_{i=1}^{n} I(x_i - x'_i, \delta') \tag{21}$$

where $I(x_i - x'_i, \delta')$ is an indicator function.

$$I(|x_i - x'_i|, \delta') = \begin{cases} 1, & |x_i - x'_i| > \delta' \\ 0, & |x_i - x'_i| < \delta' \end{cases} \tag{22}$$

The final threshold δ could be chosen by percentage criterion on the value of overall reconstruction error. It is determined by the value of the array of reconstruction error ordered in descent at a given percentage.

Once the threshold δ is determined, we could select part of the training set, in which the reconstruction error exceeds δ, as the training set for the boosting model. The boosting model can be implemented by many existing classification approaches, such as Decision Tree (DT), Support Vector Machine (SVM), and Single hidden layer Neural Network (SNN). It is also feasible to use another deep network as the boosting model.

The boosting model is trained by samples whose reconstruction error exceeds the threshold δ. The training set for the boosting model is a subset of the training set for the deep network. It can produce another output vector $Y'' = \{y''_1, y''_2, \ldots, y''_m\}$.

D. Result Averaging

The last step in the dynamic boosting strategy is to average the result output by the deep network model and the alternative model. For the testing samples whose reconstruction error do not exceeds the threshold, the result output by the deep network is the final result, i.e., $Y_o = Y'$. For other testing samples, the final result is the average result as $Y_o = (Y' + Y'')/2$. We could also use the output of the boosting model as the result.

The boosting strategy is some kind of similar with the traditional Bagging strategy [19]. In Bagging, different models are trained on different random selections of cases from the training set and all models are given equal weight in the combination. The difference between our boosting strategy and Bagging is that the dataset for boosting is determined by the reconstruction error instead of random selection.

The boosting strategy is dynamic because we rely on reconstruction error of each testing sample to determine whether we need to use the boosting model or not. The reconstruction error is produced in real time with the output of the classification model.

It is also worth to notice that the dimension of reconstruction error is as the same as the input vector X. If we use the deep network as the boosting model, we can stack several deep networks together, each relies on the reconstruction error produced by the previous deep network.

V. Experimental Results

A. Experiment Settings

Datasets used. Three datasets are used in the experiments to test our approach, namely: *MNIST:* the well-known digit classification problem and *CIFAR:* the CIFAR-10 image-classification task. *Air quality dataset:* hourly reported air quality index of a middle sized city in China of two months is collected in the dataset. The air quality index includes index of six main pollutants in the air. Our task is predicting index of a pollutant would exceed the threshold in the next several hours.

Threshold δ. We tested the threshold δ for reconstruction error from 10%-100% value of overall reconstruction error of the training sample with 10% as a gap. All the training samples are used to train the boosting model when $\delta = 100\%$. The best threshold for each comparing model may be different due to the nature of the boosting model.

Comparing methods. In our experiments, we compared two deep networks and several boosting models as follows:

- RBM: only one deep network implemented as Stacked Restricted Boltzmann Machine.
- RBM+DT: RBM as the deep network and Decision Tree as the boosting model.
- RBM+KNN: RBM as the deep network and K-Nearest Neighbors as the boosting model.

- RBM+SVM: RBM as the deep network and Support Vector Machine as the boosting model.
- RBM+SNN: RBM as the deep network and single hidden layer Neural Network as the boosting model.
- RBM+RBM: RBM as the deep network and another RBM as the boosting model.
- AE: only one deep network implemented as basic Stacked Auto Encoders.
- AE+DT: AE as the deep network and DT as the boosting model.
- AE+KNN: AE as the deep network and KNN as the boosting model.
- AE+SVM: AE as the deep network and SVM as the boosting model.
- AE+SNN: AE as the deep network and SNN as the boosting model.
- AE+AE: AE as the deep network and another AE as the boosting model.

B. *Reconstruction Error vs Classification Error*

In this section, we show the relationship between reconstruction error and classification error to validate our claim that the prediction interval becomes bigger with the increase of classification error. Notice that predicted classification result is a vector. We compute the classification error by the following:

$$e = y_i - y_i', y_i = 1 \tag{23}$$

where $y_i = 1$ is the real class label. If $e < 0.5$, the classification result is correct. It is still possible that the result is correct when $e > 0.5$. For example, if $y_1' = 0.4, y_2' = 0.3, y_3' = 0.3$ and $y_1 = 1$, the error is 0.6 while the classification result is correct.

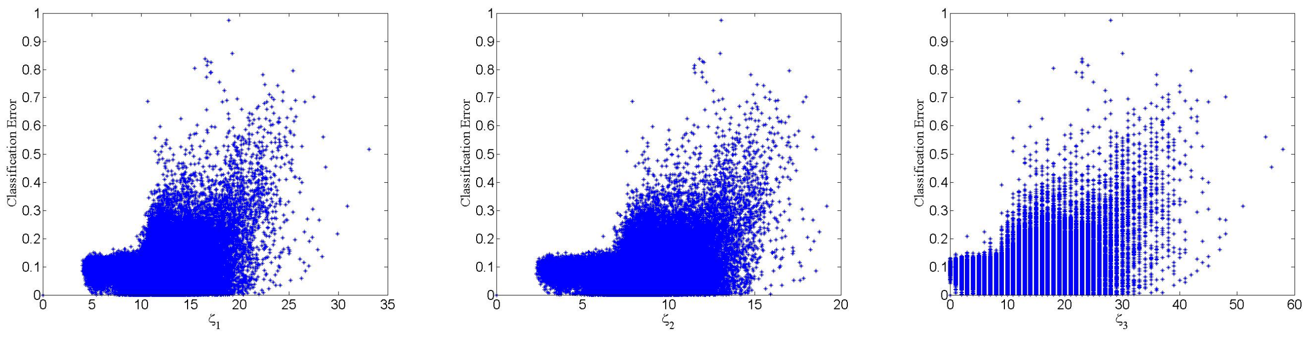
We recorded the classification error and reconstruction error on all $60,000$ training samples in MNIST dataset. Three different ways $(\zeta_1, \zeta_2, \zeta_3)$ are tried to computed the overall reconstruction error as introduced in previous section. The top-k for ζ_2 is set at 100 while there are $28 \times 28 = 784$ inputs. The threshold δ' for ζ_3 is set at 0.2. The results are reported in Figure 5. From the figure, we can see that most of the samples are with small reconstruction error. The points with small overall reconstruction error are dense while the points with big reconstruction error are sparse in the figure. For those samples with small overall reconstruction error, the classification error is small either. It supports our analysis on prediction interval of deep learning and its relationship with reconstruction error and classification error. Though not all the samples with big reconstruction error are misclassified, the average prediction interval is bigger for those samples.

Figure 5 (a) and Figure 5 (b) are actually very similar. As mentioned, ζ_1 is a special case of ζ_2 when the top-k inputs are all the inputs. The top-100 reconstruction error is more than half in overall reconstruction error from the figure while it is only $100/784 \approx 13\%$ in amount of inputs. It implies that most of the input features can be reconstructed with small error while only part of the features are reconstructed incorrectly. These inaccurately reconstructed features are the key to the final results since the distributions of Figure 5 (a) and Figure 5 (b) are similar. It also supports the claim that reconstruction error is highly correlated with classification error.

C. *Performance Comparison*

In this section, we give comparisons of performance of considered models on two datasets. For the network parameter of the basic RBM and AE we refer to the setting in the work of Hinton and Bengio [9][2]. The final performance is related with the threshold δ. The threshold δ determines which part of the training samples are used to train the boosting model and how many testing samples in the testing set should be reclassified.

We give the test error of considered methods on MNIST dataset under different setting of δ in Table 1. Two most widely used deep networks: RBM and AE, are used for the deep network model. Several models are implemented as the boosting model. Here, the threshold δ is the level of reconstruction error in training samples. Value of overall reconstruction error equals the top 10%th value when ordering the reconstruction error of training samples in descent when $\delta = 0.1$. No boosting strategy is used when $\delta = 0$ and all the samples are used to train the boosting model when $\delta = 1$. From the table, we can see that no matter what the



(a) (b) (c)

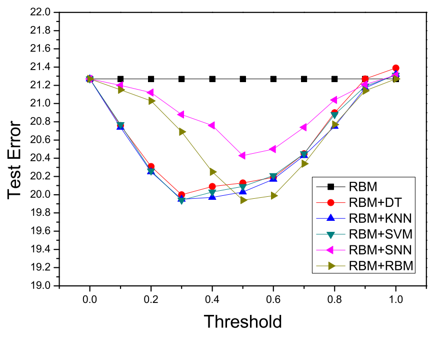
Fig. 5: Relationship between reconstruction error and classification error.

TABLE I: Performance comparison of the considered models on MNIST with the variation of the threshold. Test error is reported. Best performer in each model is in bold.

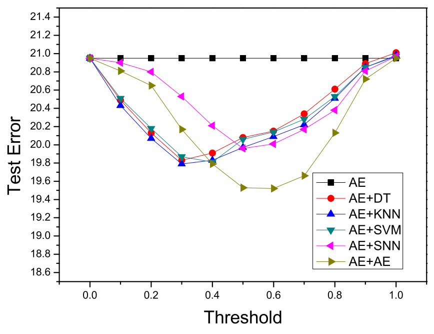
Method	Threshold										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RBM	1.52	1.52	1.52	1.52	1.52	1.52	1.52	1.52	1.52	1.52	1.52
RBM+DT	1.52	1.48	1.46	**1.4**	1.41	1.43	1.43	1.45	1.48	1.52	1.54
RBM+KNN	1.52	1.47	1.45	1.39	**1.38**	1.4	1.42	1.44	1.47	1.51	1.53
RBM+SVM	1.52	1.47	1.44	**1.37**	1.4	1.41	1.41	1.45	1.48	1.52	1.53
RBM+SNN	1.52	1.51	1.49	1.45	1.41	**1.4**	1.42	1.44	1.48	1.5	1.52
RBM+RBM	1.52	1.52	1.49	1.43	1.38	**1.37**	1.41	1.43	1.46	1.49	1.52
AE	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97
AE+DT	1.97	1.93	1.88	**1.82**	**1.82**	1.83	1.86	1.9	1.94	1.97	2
AE+KNN	1.97	1.91	1.86	**1.79**	1.82	1.81	1.85	1.89	1.92	1.96	1.99
AE+SVM	1.97	1.91	1.85	**1.78**	1.8	1.83	1.86	1.9	1.93	1.97	1.98
AE+SNN	1.97	1.96	1.93	1.9	1.86	**1.85**	1.87	1.89	1.93	1.95	1.97
AE+AE	1.97	1.95	1.92	1.84	1.78	**1.77**	1.81	1.84	1.88	1.92	1.97

boosting model is, our boosting strategy is effective. Only in some situations when the threshold is near 1, the performance of the boosting model is worse than the raw deep model. When the threshold is setting at 1, all the training samples are used to train the boosting model. Therefore, the result is the averaging of two classification models. As reported in existing work of deep learning, deep network is more effective than traditional classification models such as DT and KNN. The average result is worse than the single deep network. The boosting strategy is useful in other situations. Most of the boosting models obtain the best performance when $\delta \in [0.3, 0.4]$. When δ is small, only limited number of testing samples are reclassified using the boosting model. Many misclassified testing samples are not boosted because of the small δ. When δ is big, the boosting model can not take full advantage of reconstruction error because many samples with small reconstruction error are also included in the training set of the boosting model. For the testing samples, many samples are correctly classified by the deep network would be reclassified by the boosting model again due to the big δ. It may lead some correctly classified samples being misclassified when considering the average results of the deep network and the boosting model. Hence, the best performance is obtained when δ is around 0.3 to 0.4. It is a little bit different when the deep network is used as the boosting model. As we know, the deep network as well as the neural network requires large amount of samples to reach good performance because its complex structure. Therefore, the best δ is around 0.5 for the deep network as the boosting model. The improvements for different models are similar. The boosting strategy can reduce the testing error about 0.15.

The comparison of performance on CIFAR dataset is reported in Figure 6. The results on CIFAR are similar with the results on MNIST as demonstrated in the figure. All the boosting models are effective in classification. It implies that we can use reconstruction error to improve the performance of the model with appropriate parameter settings. The best threshold is around 0.3 for approaches using DT, KNN, SVM as the boosting model while it is 0.5 for dual deep networks.



(a) RBM Family



(b) AE Family

Fig. 6: Performance comparison of the considered models on CIFAR.

Finally, we report the experimental result on air quality dataset in Figure 7. The task of predicting whether the index would exceed the threshold is a binary classification task. We use the data of the first 18 months as the training data and the data of the rest 6 months as the testing set. Input data is the index of six pollutants in previous 24 hours (144 in dimension). The predicting time step ranges from 1 hour to 24 hours. For the sake of simplicity, we only report the result comparison of AE and AE+AE. The threshold is set at 0.5. From the figure, we can see that our boosting strategy is effective in predicting. It could reduce the classification error over 1%. With the increase of predicting time step, the classification error increases as well. However, the advantage of the boosting method is more obvious with the increase of predicting time step. When the predicting time step is small, the reconstruction error is small as well. The boosting strategy would not take effect in this situation.

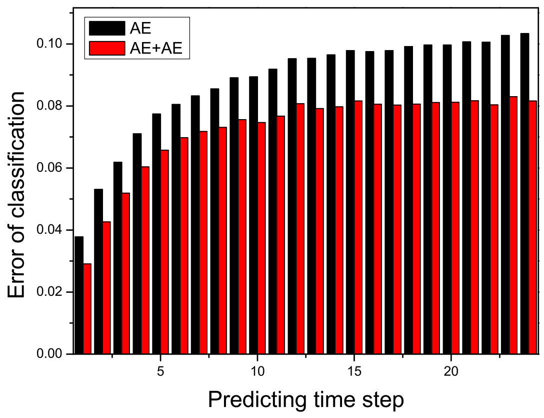


Fig. 7: Classification error on air quality dataset.

In conclusion, from the experiments, we validated that we could use reconstruction error to improve the performance of the classification using our boosting strategies. In addition, appropriate threshold δ is the key to the good performance of the final results. We can determine the value of the threshold from the cross-validation of boosting strategies in training set.

VI. Conclusion

In this paper, we proposed a dynamic boosting strategy according to reconstruction error in deep networks. We first analyzed the relationship between reconstruction error and classification error. From the perspective of prediction interval, we demonstrated that with the increase of reconstruction error, the prediction interval would become bigger. Therefore, the classification result is not reliable when the reconstruction error is big. Then we proposed a boosting strategy based on the reconstruction error and the classification error in the training set. With an appropriate threshold δ, we could select part of the training set to learn a boosting model. We can record the reconstruction error of the selected training samples and use them as inputs for the boosting model. From experiments on two widely used classification datasets, we validated that our boosting strategy is effective in improving the performance of classification.

There are still many potential works to do of boosting using reconstruction error in deep learning. It is ideal that we can directly modify the structure of the deep network to improve the performance instead of relying on an extra boosting model. Reconstruction error may provide us useful information about the parameter we learned. So it is possible to directly modify the weight in the deep network according to reconstruction error. Another interesting direction is exploring the theoretical bound of the boosting strategy. It may rely on the theoretical bound of deep learning. We also want to test the dynamic boosting strategy for other tasks such as regression. The boosting strategy is data driven so that it is valuable to real-time tasks such as regression in time series data.

References

[1] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[4] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] Richard D De VIEAUX, Jennifer Schumi, Jason Schweinsberg, and Lyle H Ungar. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[6] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on Artificial Intelligence and Statistics*, pages 153–160, 2009.

[7] Tom Heskes. Practical confidence and prediction intervals. *Advances in neural information processing systems*, pages 176–182, 1997.

[8] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[9] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[11] JT Gene Hwang and A Adam Ding. Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757, 1997.

[12] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *Neural Networks, IEEE Transactions on*, 22(9):1341–1356, 2011.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[14] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[15] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40, 2009.

[16] Arnold Ludovic, Paugam-Moisy Hélene, and Sebag Michele. Unsupervised layer-wise model selection in deep neural networks. 2010.

[17] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[18] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 1, pages 55–60. IEEE, 1994.

[19] J Ross Quinlan. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[20] Durga L Shrestha and Dimitri P Solomatine. Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2):225–235, 2006.

[21] Yee Whye Teh and Geoffrey E Hinton. Rate-coded restricted boltzmann machines for face recognition. *Advances in neural information processing systems*, pages 908–914, 2001.

[22] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 9999:3371–3408, 2010.